Dress like a Star: Retrieving Fashion Products from Videos

Noa Garcia
Aston University, UK
garciadn@aston.ac.uk

George Vogiatzis
Aston University, UK
g.vogiatzis@aston.ac.uk

Abstract

This work proposes a system for retrieving clothing and fashion products from video content. Although films and television are the perfect showcase for fashion brands to promote their products, spectators are not always aware of where to buy the latest trends they see on screen. Here, a framework for breaking the gap between fashion products shown on videos and users is presented. By relating clothing items and video frames in an indexed database and performing frame retrieval with temporal aggregation and fast indexing techniques, we can find fashion products from videos in a simple and non-intrusive way. Experiments in a large-scale dataset conducted here show that, by using the proposed framework, memory requirements can be reduced by 42.5X with respect to linear search, whereas accuracy is maintained at around 90%.

1. Introduction

Films and TV shows are a powerful marketing tool for the fashion industry, since they can reach thousands of millions of people all over the world and impact on fashion trends. Spectators may find clothing appearing in movies and television appealing and people's personal style is often influenced by the multimedia industry. Also, online video-sharing websites, such as YouTube[1], have millions of users generating billions of views every day[2] and famous *youtubers*[3] are often promoting the latest threads in their videos.

Fashion brands are interested in selling the products that are advertised in movies, television or YouTube. However, buying clothes from videos is not straightforward. Even when a user is willing to buy a fancy dress or a trendy pair of shoes that appear in the latest blockbuster movie, there is often not enough information to complete the purchase. Finding the item and where to buy it is, most of the times, difficult and it involves time-consuming searches.

Figure 1: Fashion items in a popular TV show.

To help in the task of finding fashion products that appear in multimedia content, some websites, such as *Film Grab*[4] or *Worn on TV*[5], provide catalogs of items that can be seen on films and TV shows, respectively. These websites, although helpful, still require some effort before actually buying the fashion product: users need to actively remember items from videos they have previously seen and navigate through the platform until they find them. On the contrary, we propose an effortless, non-intrusive and fast computer vision tool for searching fashion items in videos.

This work proposes a system for retrieving clothing products from large-scale collections of videos. By taking a picture of the playback device during video playback (i.e. an image of the cinema screen, laptop monitor, tablet, etc.), the system identifies the corresponding frame of the video sequence and returns that image augmented with the fashion items in the scene. In this way, users can find a product as soon as they see it by simple taking a photo of the video sequence where the product appears. Figure 1 shows a frame

[1] https://www.youtube.com/
[2] https://www.youtube.com/yt/press/statistics.html
[3] Users who have gained popularity from their videos on YouTube
[4] http://filmgarb.com/
[5] https://wornontv.net

of a popular TV show along with its fashion items.

Recently, many computer vision applications for clothing retrieval [14, 11] and style recommendation [24, 21] have been proposed. Our system is close to clothing retrieval approaches in that the aim is to retrieve a product from an image. Instead of retrieving products directly as in standard clothing retrieval, we propose to first retrieve frames from the video collection. There are several reasons for that. Firstly, in standard clothing retrieval users usually provide representative images of the object of interest (e.g. dresses in front view, high-heeled shoes in side view, etc.). In a movie, the view of the object of interest cannot be chosen, and items might be partially or almost completely occluded, such as the red-boxed dress in Figure 1. Secondly, standard clothing retrieval requires to select a bounding box around the object of interest. This is undesirable in clothing video retrieval as it may distract user's attention from the original video content. Finally, performing frame retrieval instead of product retrieval in videos allows users to get the complete list of fashion items in a scene, which usually matches the character's style, including small accessories such as earrings, watches or belts. Some of these items are usually very small, sometimes almost invisible, and would be impossible to detect and recognize using standard object retrieval techniques.

Retrieving frames from videos is a challenging task in terms of scalability. An average movie of two hours duration may contain more than 200,000 frames. That means that with only five movies in the video dataset, the number of images in the collection might be over a million. To overcome scalability issues, we propose the combination of two techniques. Firstly, frame redundancy is exploited by tracking and summarizing local binary features [7, 20] into a *key feature*. Secondly, key features are indexed in a kd-tree for fast search of frames. Once the scene the query image belongs to is identified, the clothing items associated to that scene are returned to the user.

The contributions of this paper are:

- Introduction of the video clothing retrieval task and collection of a dataset of videos for evaluation.
- Proposal of a fast and scalable video clothing retrieval framework based on frame retrieval and indexing algorithms.
- Exhaustive evaluation of the framework, showing that similar accuracy to linear search can be achieved while the memory requirements are reduced by a factor of 42.5.

This paper is structured as follows: related work is summarized in Section 2; the details of the proposed system are explained in Section 3; experiment setup and results are detailed in Section 4 and 5, respectively; finally, the conclusions are summarized in Section 6.

2. Related Work

Clothing Retrieval. Clothing retrieval is the field concerned with finding similar fashion products given a visual query. In the last few years, it has become a popular field within the computer vision community. Proposed methods [14, 25, 13, 11, 12, 15] are mainly focused on matching real-world images against online catalog photos. In this cross-scenario problem, some methods [25, 13, 15] train a set of classifiers to find items with similar attributes. Other approaches [14, 12, 11] model the differences across domains by using a transfer learning matrix [14] or a deep learning architecture [12, 11]. All these methods, however, perform clothing retrieval on static images, in which a representative and recognizable part of the item of interest is contained. In the scenario of clothing retrieval from films and television, fashion items are not always shown in a representative and recognizable way across the whole duration of the video due to object occlusions and changes in camera viewpoints. Given a snapshot of a video, an alternative solution is to first, identify the frame it belongs to and then, find the products associated to it.

Scene Retrieval. Scene retrieval consists on finding similar frames or scenes from a video collection according to a query image. Early work, such as Video Google [22] and others [18, 9, 8], retrieve frames from video datasets by applying image retrieval methods and processing each frame independently. More recent scene retrieval approaches use temporal aggregation methods to improve scalability and reduce memory requirements. [2, 5] extract hand-crafted local features from frames, track them along time and aggregate them into a single vector. Other proposals [26, 3, 4] produce a single scene descriptor, which improves efficiency but involves a more difficult comparison against the static query image. Recently, Araujo *et al.* [4] showed that, in scene retrieval, methods based on hand-crafted features outperform convolutional neural networks.

Similarly to [2], our approach is based on the aggregation of local features along temporal tracks. Instead of using SIFT features [16], we provide evidence that binary features [7, 20] are more appropriate for the task and that they can be easily indexed in a kd-tree for fast search.

3. Video Clothing Retrieval Framework

To retrieve fashion products from movies, television and YouTube, we propose a framework based on frame retrieval and fast indexing techniques. Since the number of frames explodes with the number of movies available in a collection, a key feature of this framework is its scalability. The system overview is shown in Figure 2. There are three main modules: product indexing, training phase and query phase. The product indexing and the training phase are done offline, whereas the query phase is performed online.

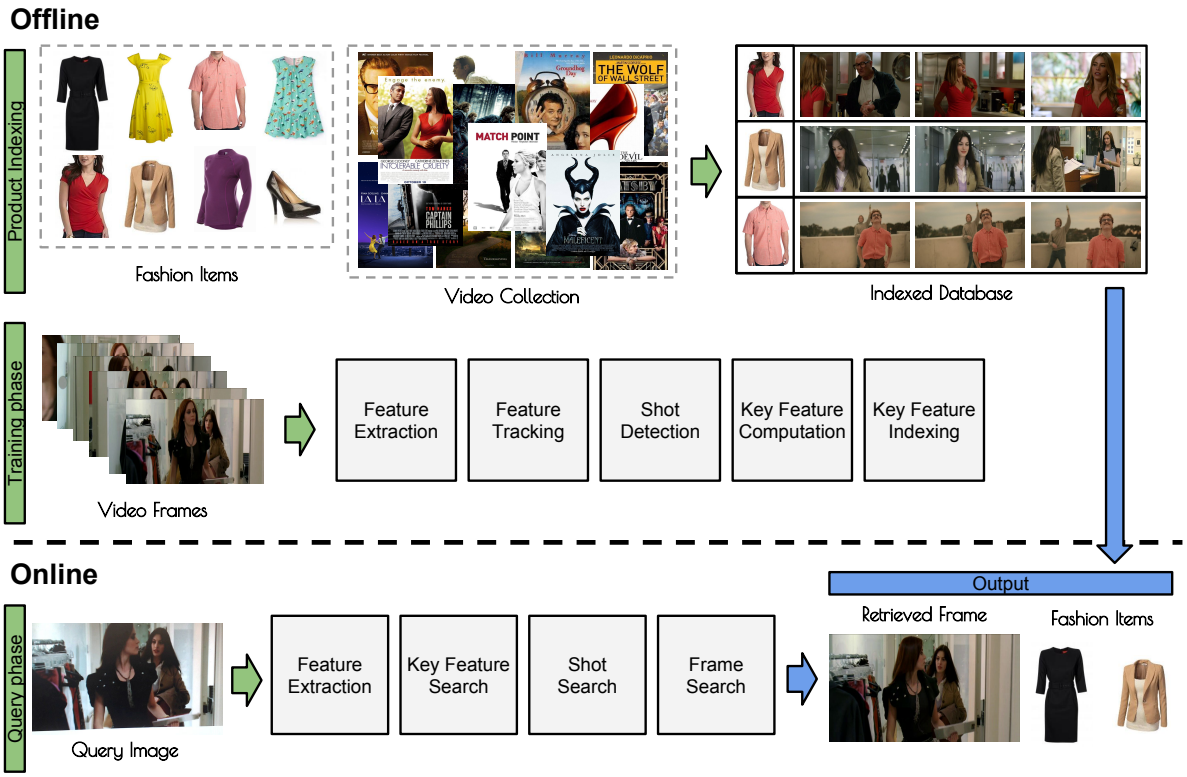


Figure 2: System overview. First, during the product indexing, fashion items and frames are related in an indexed database. Then, frames are indexed by using *key features*. Finally, in the query phase, query images are used to retrieve frames and find associated fashion items.

In the product indexing, fashion items and frames from the video collection are related in an indexed database. This process can be done manually (e.g. with Amazon Mechanical Truck[6]) or semi-automatically with the support of a standard clothing retrieval algorithm. In the training phase, features are extracted and tracked along time. Tracks are used to detect shot boundaries and compute aggregated key features, which are indexed in a kd-tree. In the query phase, features extracted from the query image are used to find their nearest key features in the kd-tree. With those key features, the shot and the frame the query image belong to are retrieved. Finally, the fashion products associated with the retrieved frame are returned by the system.

3.1. Feature Extraction and Tracking

Local features are extracted from every frame in the video collection and tracked across time by applying descriptor and spatial filters. The tracking is performed in a bidirectional way so features within a track are unique (i.e. each feature can only be matched with up to two features: one in the previous frame and one in the following frame).

As local features, instead of the popular SIFT [16], we use binary features for two main reasons. Firstly, because Hamming distance for binary features is faster to compute than Euclidean distance for floating-points vectors. Secondly, because we find that binary features are more stable over time than SIFT, as can be seen in Figure 3. Convolutional neural network (CNN) features from an intermediate layer of a pre-trained network were also studied. However, their results were not satisfactory as the resulting tracked features quickly started to get confused when the dataset increased.

3.2. Shot Detection and Key Feature Computation

Consecutive frames that share visual similarities are grouped into shots. The boundaries of different shots are detected when two consecutive frames have no common tracks. Each shot contains a set of tracks, each track representing the trajectory of a particular feature along time. We define a *key feature* as the aggregation of all the features in the same track into a single vector. Subsequently, each shot is then represented by a set of key features, similarly to how frames are represented by a set of features. For each track, a key feature is computed by using majorities [10].

[6]https://www.mturk.com

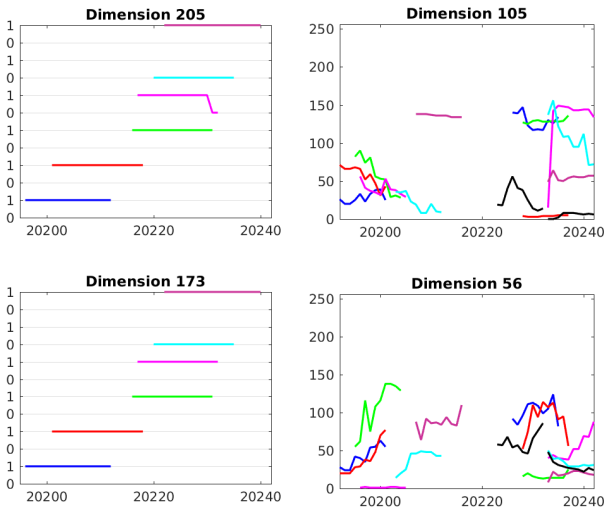


Figure 3: Trajectories of sample tracks along a sequence of frames. Left: Binary features. Right: SIFT features. Binary features are more constant over time than SIFT features.

3.3. Key Feature Indexing

A popular method for searching in binary feature space is FLANN [17]. FLANN uses multiple, randomly generated hierarchical structures and it is not obvious how it can be deployed across multiple CPUs in order to provide the scalability we are looking for. On the other hand, kd-trees are a formalism that has been shown to be highly parallelizable in [1]. In this work, we modify the basic kd-tree to handle binary features and index our key features.

In a kd-tree each decision node has an associated dimension, a splitting value and two child nodes. If the value of a query vector in the associated dimension is greater than the splitting value, the vector is assigned to the left child. Otherwise, the vector is associated to the right child. The process is repeated at each node during the query phase until a leaf node is reached. In our case, as we are dealing with binary features, each decision node has an associated dimension, dim, such that all query vectors, v, with $v[dim] = 1$ belong to the left child, and all vectors with $v[dim] = 0$ belong to the right child. The value dim is chosen such that the training data is split more evenly in that node, i.e. its entropy is maximum. Note that this criterion is similar to the one used in the ID3 algorithm [19] for the creation of decision trees, but where the splitting attribute is chosen as the one with smallest entropy. Leaf nodes have as many as S_L indices pointing to the features that ended up in that node. A first-in first-out (FIFO) queue keeps record of the already visited nodes to backtrack B times and explore them later. We use a FIFO queue to ensure that even if some of the bits in a query vector are wrong, the vector can reach its closest neighbours by exploring unvisited nodes latter. With the FIFO queue we first explore the closest nodes to the root, because the corresponding bits exhibit more variability by construction of the tree.

Figure 4: Visual similarities between frames. Left: Query image. Middle: Ground truth frame. Right: Retrieved frame similar to ground truth frame, thus, Visual Match.

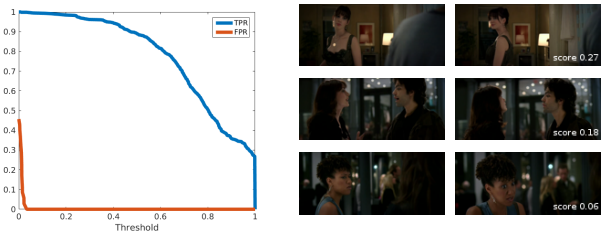
3.4. Search

In the query phase, binary features are extracted from an input image and assigned to its nearest set of key features by searching down the kd-tree. Each key feature votes for the shot it belongs to. The set of frames contained in the most voted shot are compared against the input image by brute force, i.e. distances between descriptors in the query image and descriptors in the candidate frames are computed. The frame with minimum distance is retrieved. Shots are commonly groups of a few hundreds of frames, thus the computation can be performed very rapidly when applying the Hamming distance. Finally, all the fashion vectors associated with the retrieved frame are returned to the user.

4. Experiments

Experimental Details. To evaluate our system a collection of 40 movies with more than 80 hours of video and up to 7 million frames is used. Query images are captured by a webcam (Logitech HD Pro Webcam C920) while movies are being played on a laptop screen. To provide a ground truth for our experiments, the frame number of each captured image is saved in a text file. Movies and queries have different resolutions and aspect ratios, thus all the frames and images are scaled down to 720 pixels in width. Binary features are computed by using ORB detector [20] and BRIEF extractor [7]. In the tracking module, only matches with a Hamming distance less than 20 and a spatial distance less than 100 pixels are considered, whereas in the key feature computation algorithm, only tracks longer than 7 frames are used. The default values for the kd-tree are set at $S_L = 100$ and $B = 50$.

Evaluation criteria. The aim of the system is to find the fashion products in the query scene, so we only retrieve one frame per query image. The returned frame may not be the same frame as the one in the annotated ground truth but one visually similar. As long as the retrieved frame shares strong similarities with the ground truth frame, we consider it as a *Visual Match*, as shown in Figure 4. To measure the visual similarities between two frames, we match SURF [6]



(a) (b)

Figure 5: Precision of the evaluation method: (a) TPR and FPR curves; (b) Three ground truth frames (left column) and retrieved frames (right column) along with their score.

features. The linear comparison between two frames that do not present any noise or perspective distortion is an easy task that almost all kinds of features can perform correctly. If the matching score between SURF features of two dataset frames is greater than a threshold, τ, they are considered to be visually similar.

To measure the precision of this evaluation method, we manually annotate either if a pair of frames (i.e. ground truth frame and retrieved frame) is a Visual Match or not, along with its score. For different values of τ the True Positive Rate (TPR) as well as the False Positive Rate (FPR) are computed as:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{1}$$

where TP is the number of true positives (Visual Match with score $> \tau$), FP is the number of false positives (No Match with score $> \tau$), TN is the number of true negatives (No Match with score $\leq \tau$) and FN is the number of false negatives (Visual Match with score $\leq \tau$). Figure 5 shows both the TPR and the FPR computed with the annotations of 615 pairs of frames and 406 different values of τ. Among all the possible values, we chose $\tau = 0.15$, with $TPR = 0.98$ and $FPR = 0$. Finally, the accuracy for a set of query images is computed as:

$$\text{Acc} = \frac{\text{No. Visual Matches}}{\text{Total No. Queries}} \tag{2}$$

5. Experimental Results

To evaluate the proposed framework, two different experiments are performed. In the first one, our frame retrieval system is compared against other retrieval systems. In the second one, the performance of the frame retrieval method when scaling up the video collection is evaluated in terms of accuracy.

5.1. Retrieval Performance

First, we compare our system against other retrieval frameworks.

		BF	KT	KF	Ours
Indexed Features		85M	85M	25M	2M
Memory		2.53GB	2.53GB	762MB	61MB
Accuracy	B = 10	0.98	0.90	0.91	0.92
	B = 50		0.94	0.92	0.93
	B = 100		0.96	0.93	0.94
	B = 250		0.97	0.93	0.94

Table 1: Comparison between different systems on a single movie. Accuracy is shown for four backtracking steps, B. Our system achieves comparable results by using 42.5x less memory than BF and KT and 12.5x less memory than KF.

BF. Brute Force matcher, in which query images are matched against all frames and all features in the database. Brute Force system is only used as an accuracy benchmark, since each query take, in average, 46 minutes to be processed. Temporal information is not used.

KT. Kd-Tree search, in which all features from all frames are indexed using a kd-tree structure, similarly to [1]. Temporal information is not used.

KF. Key Frame extraction method [23], in which temporal information is used to reduce the amount of frames of each shot into a smaller set of key frames. Key frames are chosen as the ones at the peaks of the distance curve between frames and a reference image computed for each shot. For each key frame, features are extracted and indexed in a kd-tree structure.

We compare these three systems along with the method proposed here, using a single movie, *The Devil Wears Prada*, consisting of 196,572 frames with a total duration of 1 hour 49 minutes and 20 seconds. To be fair in our comparison, we use binary features in each method. The results of this experiment are detailed in Table 1. The performance is similar for the four systems. However, the amount of processed data and the memory requirements are drastically reduced when the temporal information of the video collection is used. In the case of our system, by exploiting temporal redundancy between frames, the memory is reduced by 42.5 times with respect to BF and KT and by 12.5 times with respect to the key frame extraction technique. Theoretically, that means that when implemented in a distributed kd-tree system as the one in [1], where the authors were able to process up to 100 million images, our system might be able to deal with 4,250 million frames, i.e. more than

Title	N. Frames	N. Features	N. Shots	N. Key Features	N. Queries	Accuracy
The Help	210387	101M	1726	2.2M	813	0.98
Intolerable Cruelty	179234	86M	1306	2M	544	0.97
Casablanca	147483	71M	881	1.5M	565	0.96
Witching & Bitching	163069	66M	4193	0.8M	588	0.74
Pirates of the Caribbean 3	241127	108M	3695	1.7M	881	0.74
Captain Phillips	190496	59M	7578	0.6M	618	0.67
Total	**7M**	**3040M**	**116307**	**58M**	**25142**	**0.87**

Table 2: Summary of the results from the experiments on the dataset containing 40 movies. Due to space restrictions, only three of the best and worst accuracies are shown, along with the total results for the whole database.

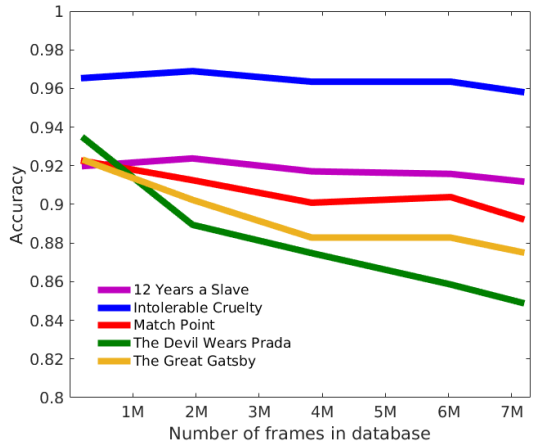


Figure 6: Accuracy vs Database size for 5 different movies.

20,000 movies and 40,000 hours of video.

5.2. Large-scale Dataset

In this experiment, we explore the scalability of our framework by increasing the dataset up to 40 movies and 7 million frames. The collection is diverse and it contains different movie genres such as animation, fantasy, adventure, comedy or drama, to ensure that a wide range of movies can be indexed and retrieved with the proposed algorithm.

Table 2 shows the results of this experiment. It can be seen that, by using our temporal aggregation method, the amount of data is reduced from 7 million frames and 3,040 million features to only 116,307 shots and 58 million key features. Even so, the total number of key features in the 40 movie collection is still smaller than the 80 million features that, in average, a single movie contains. The total accuracy over the 40 movies is 0.87, reaching values of 0.98 and 0.97 in *The Help* and *Intolerable Cruelty* movies, respectively. Movies with very dark scenes such as *Captain Phillips* and *Pirates of the Caribbean 3* perform worst, as fewer descriptors can be found in those kinds of dimly lit images.

Figure 6 shows the evolution of accuracy when the size of the database increases for five different movies. It can be seen that most of the movies are not drastically affected when the number of frames in the database is increased from 200,000 to 7 million. For example, both *Intolerable Cruelty* and *12 Years a Slave* movies maintain almost a constant accuracy for different sizes of the collection. Even in the worst case scenario, *The Devil Wears Prada* movie, the loss in accuracy is less than a 8.5%. This suggests that our frame retrieval system is enough robust to handle large-scale video collections without an appreciable loss in performance.

6. Conclusions

This work proposes a framework to perform video clothing retrieval. That is, given a snapshot of a video, identify the video frame and retrieve the fashion products that appear in that scene. This task would help users to easily find and purchase items shown in movies, television or YouTube videos. We propose a system based on frame retrieval and fast indexing. Experiments show that, by using our temporal aggregation method, the amount of data to be processed is reduced by a factor of 42.5 with respect to linear search and accuracy is maintained at similar levels. The proposed system scales well when the number of frames is increased from 200,000 to 7 million. Similar to [1] our system can easily be parallelised across multiple CPUs. Therefore, the encouraging experimental results shown here indicate that our method has the potential to index fashion products from thousands of movies with high accuracy. In future work we intend to further increase the size of our dataset in order to identify the true limits of our approach. Furthermore it is our intention to make all data publicly available so that other researchers can investigate this interesting retrieval problem.

References

[1] M. Aly, M. Munich, and P. Perona. Distributed kd-trees for retrieval from very large image collections. *British Machine Vision Conference*, 2011. 4, 5, 6

[2] A. Anjulan and N. Canagarajah. Object based video retrieval with local region tracking. *Signal Processing: Image Communication*, 22(7), 2007. 2

[3] A. Araujo, J. Chaves, R. Angst, and B. Girod. Temporal aggregation for large-scale query-by-image video retrieval. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 1519–1522. IEEE, 2015. 2

[4] A. Araujo and B. Girod. Large-scale video retrieval using image queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 2

[5] A. Araujo, M. Makar, V. Chandrasekhar, D. Chen, S. Tsai, H. Chen, R. Angst, and B. Girod. Efficient video search using image queries. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3082–3086. IEEE, 2014. 2

[6] H. Bay, T. Tuytelaars, and L. Van Gool. SURF : Speeded Up Robust Features. *European Conference on Computer Vision*, pages 404–417, 2006. 4

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF : Binary Robust Independent Elementary Features. *ECCV*, 2010. 2, 4

[8] D. Chen, N.-M. Cheung, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Dynamic selection of a feature-rich query frame for mobile video retrieval. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1017–1020. IEEE, 2010. 2

[9] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 549–556. ACM, 2007. 2

[10] C. Grana, D. Borghesani, M. Manfredi, and R. Cucchiara. A fast approach for integrating orb descriptors in the bag of words model. In *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013. 3

[11] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *ICCV*, 2015. 2

[12] J. Huang, R. S. Feris, Q. Chen, and S. Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1062–1070, 2015. 2

[13] Y. Kalantidis, L. Kennedy, and L.-J. Li. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 105–112. ACM, 2013. 2

[14] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu, and S. Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3330–3337. IEEE, 2012. 2

[15] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 2

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004. 2, 3

[17] M. Muja and D. G. Lowe. Fast matching of binary features. *Proceedings of the 2012 9th Conference on Computer and Robot Vision*, 2012. 4

[18] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. *CVPR*, 2006. 2

[19] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 4

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *ICCV*, 2011. 2, 4

[21] E. Simo-Serra, S. Fidler, F. Moreno-Noguer, and R. Urtasun. Neuroaesthetics in fashion: Modeling the perception of fashionability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2015. 2

[22] J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. *ICCV*, 2003. 2

[23] Z. Sun, K. Jia, and H. Chen. Video key frame extraction based on spatial-temporal color distribution. In *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on*, pages 196–199. IEEE, 2008. 5

[24] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4642–4650, 2015. 2

[25] D. Wei, W. Catherine, B. Anurag, P.-m. Robinson, and S. Neel. Style finder: fine-grained clothing style recognition and retrieval, 2013. 2

[26] C.-Z. Zhu and S. Satoh. Large vocabulary quantization for searching instances from videos. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 52. ACM, 2012. 2